УДК 004.42:615.47

РЕАЛИЗАЦИЯ АЛГОРИТМА «ВОЛШЕБНАЯ ПАЛОЧКА» ПОД ОС ANDROID

[1]Югфельд И.Д.

[1]ФГАОУ ВПО Уральский Федеральный Университет имени первого Президента России Б.Н. Ельцина, Екатеринбург, Россия (620002 Россия, г. Екатеринбург, ул. Мира, 19)), e-mail: irinaugfeld@gmail.com

Аннотация: Настоящая статья посвящена исследованию способов реализации алгоритма «волшебная палочка» под ОС Android. Наиболее подходящий метод был получен при помощи модификации канонической реализации алгоритма с использованием рекурсии. Измененный алгоритм «волшебная палочка» построен на основе волнового алгоритма.

Ключевые слова: алгоритм «волшебная палочка», ОС Android, волновой алгоритм, распознавание образов, выделение объектов.

IMPLEMENTATION OF “MAGIC WAND” ALGORITHM FOR OS ANDROID

[1]Yugfeld I.D.

[1]Ural Federal University named after the first President of Russia B.N.Yeltsin, Ekaterinburg, Russia (Russia 620002, Ekaterinburg, Mira st., 19), e-mail: irinaugfeld@gmail.com

Abstract: This article is dedicated to the study of ways to implement the "magic wand" algorithm for OS Android. The most suitable method has been obtained by modifying the canonical algorithm implementation that use recursion. The modified "magic wand" algorithm is based on the wave algorithm.

Key words: "magic wand" algorithm, OS Android, wave algorithm, pattern recognition, object selection.

Введение

При обработке изображение часто рассматривается не как "изображение в целом", а как "изображение чего-то на фоне чего-то". [1] В общем случае при работе с тем или иным изображением часто возникает необходимость отделить одну, значимую для пользователя часть, которая интересует его в данный момент (объект), от всего остального (фон). Алгоритм «волшебная палочка» позволяет это сделать. Его особенностью является то, что пользователь может определить, что является интересующим его объектом, а что относится к фону. В данной статье будут рассмотрены особенности реализации этого алгоритма на базе ОС Android.

Идея алгоритма

Пользователь указывает точку внутри выделяемого объекта на изображении. Алгоритм определяет цвет выбранного пикселя и анализирует соседние точки на предмет совпадения цветов. Так как в реальной жизни присутствует свет, тень, отражение от

соседних предметов, то цвет объекта не однороден. Поэтому задается максимально допустимое отклонение цвета, при помощи которого и определяется принадлежность пикселя детектируемому объекту.

Реализация алгоритма

Алгоритм определения нужных пикселей является алгоритмом закрашивания замкнутых областей. Основная идея заключается в получении исходной точки (стартовой) и анализе соседних точек из 4-связной области на предмет похожести. Если точка удовлетворяет условию, то сохраняем ее в маске. Псевдокод для этого алгоритма:

```
void select(float x, float y, Color colorToMatch){
    Color color = GetColor(x, y);
    if ( colorsSimilar(color, colorToMatch) ) {
        setPixelInMask(x, y);
        select(x - 1, y, colorToMatch);
        select(x + 1, y, colorToMatch);
        select(x, y - 1, colorToMatch);
        select(x, y + 1, colorToMatch);
    }
}
```

В таком виде алгоритм предполагает применение рекурсии. Рекурсия весьма полезный инструмент, но его применение подразумевает использование практически не контролируемого ресурса: стека вызовов. А, учитывая, что параметры картинки (высота и ширина в пикселах), а также площадь выделяемого объекта в пикселах могут быть заранее не известными, то применение рекурсии может с большой вероятностью привести к переполнению стека вызовов или захватить слишком много оперативной памяти. Поэтому лучше избежать рекурсии, в данном случае это удастся при использовании идеи волнового алгоритма. Волновой алгоритм используется для поиска пути между некими точками. Суть волнового алгоритма состоит в следующем:

1. Из начального элемента распространяется в 4-х направлениях волна [2], как показано на рисунке 1. Элемент, в который пришла волна образует фронт волны. На рисунках цифрами обозначены номера фронтов волны. Каждый элемент первого фронта волны является источником вторичной волны. Элементы второго фронта волны генерируют волну третьего фронта и т.д. Процесс продолжается до тех пор, пока не будет достигнут конечный элемент. Ну, или пока не станет ясно, что его не достигнуть.

2. Строится сама трасса. Её построение осуществляется от конечного элемента к начальному.

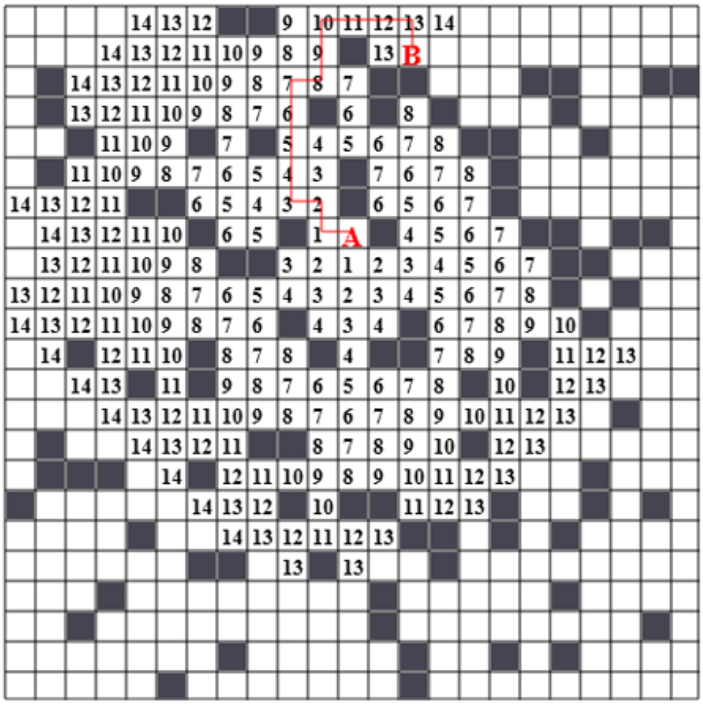

Рисунок 15. Иллюстрация работы волнового алгоритма.

Так как в текущем случае не нужно искать наименьшее расстояние от начальной точки, до конечной, то нет смысла цифрами с расстоянием помечать точки (пиксели). Вместо этого пиксель отмечается как удовлетворяющий/ неудовлетворяющий критерию вхождения в выделяемую область. Алгоритм останавливается, когда вокруг выделяемого объекта образуется замкнутая рамка из пикселей, не отвечающих установленному критерию. Псевдокод алгоритма основной части:

```
while (!next.isEmpty()){
        replaceQueuees();
        for( i=0;i<current.size();i++ ){
                checkNeighbors ( current.poll() );
        }
}
```

current, next - очереди из точек текущего и следующего уровней соответственно (На рисунке 1, если в очереди current находятся точки с пометкой 1, то в очереди next находятся точки с пометкой 2 и так далее). Псевдокод метода проверки соседних точек:

```
void checkNeighbors (Point p){
                checkForSimilarity( p, new Point(p.x, p.y-1) );
                checkForSimilarity( p, new Point(p.x, p.y+1) );
                checkForSimilarity( p, new Point(p.x-1, p.y) );
                checkForSimilarity( p, new Point(p.x+1, p.y) );
}
```

Псевдокод метода проверки пикселей на схожесть:

```
void checkForSimilarity( Point startPoint, Point checkPoint ){
        if ( isPixelExists(checkPoint) ){
                if ( areColorsSimilar( img.getPixel(startPoint.x, startPoint.y),
                                        img.getPixel(checkPoint.x, checkPoint.y) ) ){
```

```
                    mask.put( checkPoint, SIMILAR );
                    next.add( checkPoint );
                }
                else mask.put( checkPoint, NOT_SIMILAR );
            }
}
```

Псевдокод метода проверки пикселей на схожесть по цвету:

```
boolean areColorsSimilar(int c1, int c2){
int differenceR = Abs( c1.red -c2.red );
        int differenceG = Abs( c1.green -c2.green );
        int differenceB = Abs( c1.blue -c2.blue );
        int maxDifference = Max( differenceR, differenceG, differenceB );
        return maxDifference < step ? true : false;
}
```

step – чувствительность алгоритма.

Выводы

Модификация «волшебной палочки» с использованием волнового алгоритма позволила устранить рекурсию. Отказ от рекурсии исключил возможность переполнения стека вызовов, а в Android по умолчанию его размер составляет 8 Кбайт (около 260 вызовов). То есть при этом алгоритм с рекурсией будет способен распознать объект с приближенной площадью не более 260 пикселей, а это довольно небольшая цифра. Таким образом созданный алгоритм является более предсказуемым и стабильным нежели рекурсивная «волшебная палочка».

Список литературы

1. А.А. Хропов, А.С. Карпов, В.С. Лемпицкий, Д.В. Иванов, Е.П. Кузьмин. Алгоритмические основы растровой графики [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/studies/courses/993/163/info
2. Обход препятствий: волновой алгоритм (Алгоритм Ли) [Электронный ресурс]. – Режим доступа: http://suvitruf.ru/2012/05/13/1176/
3. Н.Кристофидес. Теория графов. Алгоритмический подход. М.: Мир, 1978, 432 стр.
4. Л.П. Ярославский. Введение в цифровую обработку изображений. – М.: Сов. радио, 1979. – 312 с.
5. Andy Finnell / How to implement a magic tool [Электронный ресурс]. – Режим доступа: http://www.losingfight.com/blog/2007/08/28/how-to-implement-a-magic--tool/